Videojuegos

Curso de Diseño y Programación

Nº 16 5,99 euros

Inteligencia Artificial del juego

Cómo crear el terreno

Realizar un vídeo de presentación



16

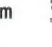






Curso práctico de Videojuegos

AUTOR DE LA OBRA
Marcos Medina
DIRECCIÓN EDITORIAL
Eduardo Toribio
etoribio@iberprensa.com
COORDINACIÓN EDITORIAL
Eva-Margarita García
eva@iberprensa.com
DISEÑO Y MAQUETACIÓN
Antonio Gª Tomé
PRODUCCIÓN
Marisa Cogorro
SUSCRIPCIONES
Tel: 91 628 02 03
Fax: 91 628 09 35
suscripciones@iberprensa.com
FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: Gráficas Don Bosco
DUPLICACIÓN CD.ROM: M.P.O.
DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Iberprensa
www.iberprensa.com


CONSEJERO
Carlos Peropadre

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, 7 (Pol. Ind. "El Nogal")
28110 Algete (Madrid)
Tel.: 91 628 02 03
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-35934-2002
ISBN: Coleccionable: 84 932417 2 5
Tomo 2: 84 932417 4 1
Obra Completa: 84 932417 5 X

Aprende divirtiéndote

Bienvenidos de nuevo a **Programación y Diseño de Videojuegos**, la primera obra coleccionable cuyo objetivo es formar al alumno en las principales técnicas relacionadas en el desarrollo completo de un videojuego.

A lo largo de la obra el lector está aprendiendo programación a nivel general y a nivel específico con ciertas herramientas y lenguajes, aprendiendo a trabajar con aplicaciones de retoque de imagen y también de diseño 3D y animación. Estamos descubriendo las aplicaciones profesionales más importantes de audio y conociendo la historia de lo que se denomina "la industria del videojuego", los últimos 20 años, los juegos que marcaron un avance, sus creadores y en general la evolución del videojuego.

Pero además, esta obra tiene un segundo objetivo, desarrollar y potenciar la creatividad del lector, nosotros a lo largo de las diferentes entregas pondremos las bases y tú pondrás tu ingenio, tu creatividad y tu capacidad de mejorar.

Nos encontramos a mitad de camino del viaje de 20 semanas que os proponemos, viaje articulado en 400 páginas y 20 CD-ROMs cuya finalidad es proporcionar las bases mínimas para después cada uno continuar su camino.

Recuerda que para alcanzar el éxito necesitas cumplir tres condiciones: que te gusten los juegos, poseer cierta dosis de creatividad y finalmente capacidad de estudio.

Una la cumples seguro.

sumario



PARA ENCUADERNAR LA OBRA:

- *Tapas del volumen 1 disponibles en Iberprensa. Pedidos por teléfono: 916280203*
- ***Tapas del volumen 2 ya a la venta en quioscos.***
- ***Los suscriptores recibirán las tapas en su domicilio sin cargo alguno como obsequio de Iberprensa.***

SERVICIO TÉCNICO:
Para consultas, dudas técnicas y reclamaciones Iberprensa ofrece la siguiente dirección de correo electrónico: games@iberprensa.com

PETICIÓN DE NÚMEROS ATRASADOS:
El envío de números sueltos o atrasados se realizará contra reembolso del precio de venta al público más el coste de los gastos de envío. Pueden ser solicitados en el teléfono de atención al cliente 91 628 02 03

Inteligencia artificial (I) Plantas y animales

Empezamos con esta entrega el desarrollo de una de las partes del juego más interesante, sutil y, sobre todo, fundamental: la inteligencia artificial.

Básicamente, el comportamiento de los habitantes de un juego parte de una serie de premisas establecidas de antemano y que luego evolucionan durante el desarrollo de la partida. Esta evolución determina un mayor o menor grado de inteligencia y depende de la cantidad de sentencias lógicas implementadas en las rutinas de control del personaje. En "Zone of Fighters" utilizaremos unos patrones fijos para programar el comportamiento de los animales y plantas que habitan en el terreno de juego, los cuales veremos en esta entrega.

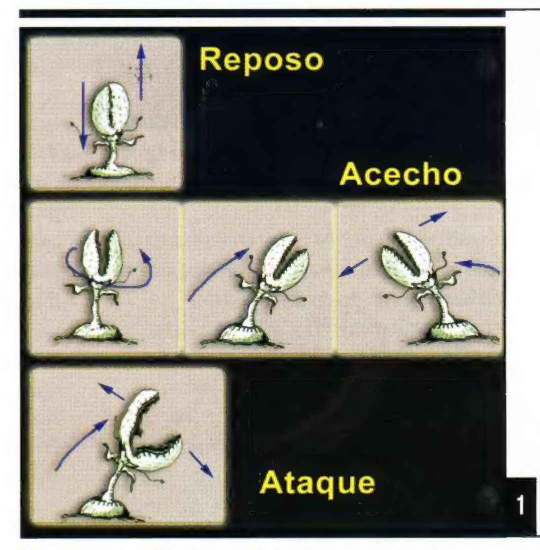
(■) DANDO VIDA A LOS ANIMALES Y PLANTAS

Antes de estudiar el comportamiento de los seres vivos debemos inicializarlos en el juego.

En el módulo "funcpantaudio.bb" situamos la función "Crear_modelos()", la cual carga e inicializa cada una de las plantas y animales del juego, así como a la propia bionave de combate y el resto de elementos del decorado. Por motivos didácticos, vamos a trabajar a partir de ahora con modelos en formato .B3D, ya que es un formato más sencillo de utilizar para el manejo de modelos con animaciones. Sin embargo, es fácil cambiar el código para utilizar modelos en formato .MD2 o .X. Para cada modelo, vamos a extraer cada una de las animaciones que forman sus movimientos. Los utilizaremos posteriormente en la función que controla su comportamiento (Ver Código 1).

Después de cargar los modelos debemos situarlos en el terreno de juego.

Como vimos en el número anterior, en la función "Crear_Decorado()" incluimos las llamadas a las funciones que colocan a los animales y plantas sobre el terreno de juego. Estas funciones están situadas en el módulo "Enemigos.bb" y funcionan exactamente igual que las desarrolladas para los edificios y otros elementos del decorado.



Diseño de los diferentes estados que toma una planta carnívora.

Código 1. Extraemos las animaciones de los modelos

```
mesh_dreeck=LoadAnimMesh("c:\zone of fighters\modelos\dreeck.b3d")
ExtractAnimSeq(mesh_dreeck,0,30)      ; reposo
ExtractAnimSeq(mesh_dreeck,31,90)     ; acechar
ExtractAnimSeq(mesh_dreeck,91,140)    ; ataque
; --------------------------------------------------------------
mesh_luny=LoadAnimMesh("c:\zone of fighters\modelos\luny.b3d")
ExtractAnimSeq(mesh_luny,0,30) ;reposo y lanzar esporas
ExtractAnimSeq(mesh_luny,0,60) ;ataque
; --------------------------------------------------------------
mesh_slunk=LoadAnimMesh("c:\zone of fighters\modelos\slunk.b3d")
ExtractAnimSeq(mesh_slunk,0,30)      ; salir de la tierra y entrar
ExtractAnimSeq(mesh_slunk,0,90)      ; ataque y a tierra
ExtractAnimSeq(mesh_slunk,80,90)     ; a tierra
```

(■) COMPORTAMIENTO DE LAS PLANTAS CARNÍVORAS

La implementación del comportamiento de los Dreecks la realizamos en la función "Actualizar_dreecks()" situada en el módulo "Enemigos.bb". Esta función tendrá que ser llamada en cada bucle de juego, por lo tanto habrá que incluirla en la función "actualizar_juego()" del módulo "Fjuego.bb":

```
Function actualizar_juego()
   ...
   actualizar_dreecks()
   actualizar_slunks()
   actualizar_lunys()
   ...
End Function
```

Diseño de los diferentes estados que toma un Luny.

Como se había diseñado, las plantas carnívoras tendrán diferentes movimientos dependiendo de su estado. Estos movimientos son los que determinan, en cierta manera, su comportamiento, y son tres: reposo, acecho y ataque.

Cada uno de estos estados se activará según ciertas reglas que programamos en la función "Actualizar_dreecks()". Esta función consta de un bucle principal que recorre cada una de las plantas creadas y que forman parte de la estructura "tipo_dreeck":

```
Function actualizar_dreecks()
    For dreeck.tipo_dreeck=
    Each tipo_dreeck
    ...
```

El control del comportamiento de la planta se basa en dos factores. Por un lado, un contador interno que determina el tiempo de un estado u otro y por otro, la proximidad de la bionave de combate. Aunque colocáramos un Dreeck en un terreno totalmente vacío debemos asignarle un comportamiento base. En él determinaremos qué movimiento debe tener. Según el diseño de esta criatura, posee la habilidad de girar y alargar el tallo para buscar y alimentarse de cualquier presa que circule en sus proximidades. Determinamos entonces una postura primaria que será de reposo y otra de acecho cuando busca alimento. Estas dos posturas se activan de forma cíclica al paso de un determinado tiempo que controlamos con el contador "tiempo_cambio" y que forma parte de la estructura de la entidad:

```
Type tipo_dreeck
  Field x_dreeck#,z_dreeck#, size#
  Field tiempo_cambio,
  tiempo_cambio_total,
  Field ataque%, distancia_comida#,
  sw_movimiento
  Field entidad_dreeck, vida_dreeck
End Type
```

Este contador es inicializado con un valor aleatorio entre 200 y 400 cuando creamos la planta en la función "Colocar_dreeck()". Cambiando este valor determinamos más o menos tiempo en cada estado:

```
dreeck\tiempo_cambio=Rnd(200,400)
dreeck\tiempo_cambio_total=
dreeck\tiempo_cambio
```

De vuelta al control del comportamiento en la función "actualizar_dreecks()", este contador irá disminuyendo en una unidad cada vez que se llama a la función:

```
dreeck\tiempo_cambio=dreeck\
tiempo_cambio-1
```

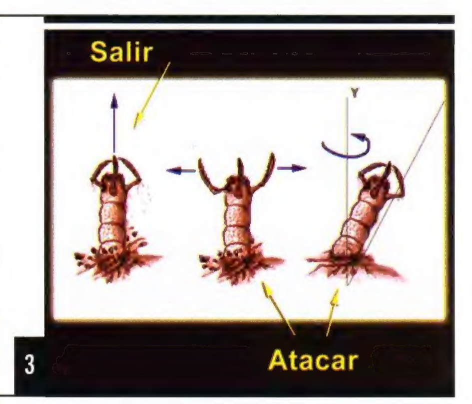
Como esta variable perderá su valor inicial, necesitamos otra auxiliar que lo conserve, ya que lo necesitaremos para algunas comparaciones y que llamamos "tiempo_cambio_total".

Básicamente, lo que haremos es que el Dreeck cambie de postura (animación) cuando este contador llegue a un determinado valor.

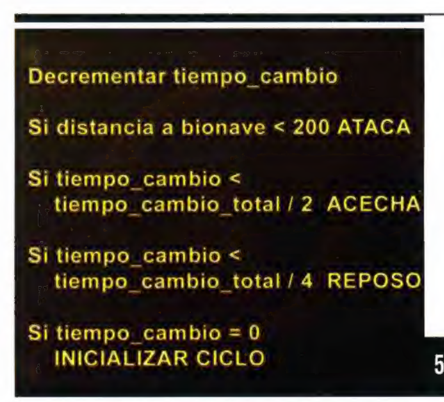
Necesitamos otras dos variables más para controlar la situación actual de la planta: un *switcher* que almacene el tipo de movimiento que está activo y que llamamos "sw_movimiento" (esta variable forma parte de la estructura de la entidad y toma dos valores: 1 ➠ planta en reposo y 2 ➠ planta en acecho) y "ataque", que indica si la planta está en ese momento en el estado de ataque (toma los valores 0 y 1) y que también forma parte de la estructura. Así, la planta estará en reposo cuando el contador "tiempo_cambio" llegué al cuarto de su valor inicial y



Utilización de "EntityDistance" para detectar la proximidad de la bionave a una planta carnívora.



Diseño de los diferentes estados que toma un Slunk.



Esquema del funcionamiento de la función que controla el comportamiento de un Dreeck.

mientras no esté atacando ("dreeck\ataque=0") (Ver Código 2).

Y cambia al estado de acecho cuando llegue a la mitad de su valor inicial (Ver Código 3).

Debemos tener en cuenta en cualquier momento de la vida del Dreeck la proximidad de la bionave de combate. Esto quiere decir que la planta cambiará al estado de ataque en el momento que divise comida. Para definir un rango de proximidad determinado a la bionave, utilizamos la instrucción "EntityDistance", la cual devuelve la distancia entre dos entidades:

```
dreeck\distancia_comida#=
EntityDistance (dreeck\
entidad_dreeck, bionave)
```

Por lo tanto, asignamos esta distancia a la variable de la entidad "dreeck\distancia_comida".

Luego aplicamos el ataque si la distancia es inferior a 200 puntos, siempre que la bionave no lleve activado el camuflaje ("camuflaje_bionave=0") y no esté atacando ya ("dreeck\ataque=0") (Ver Código 4).

Evidentemente, asignamos un 1 a la variable "ataque" para decirle a la función que la planta está en ese momento en estado de ataque. Como se puede observar, prácticamente lo que se hace es cambiar de animación. El resto lo hará el sistema de colisiones.

Para finalizar, hay que volver a inicializar todas las variables y el estado de la planta cuando el contador "tiempo_cambio" haya llegado a 0, volviendo a comenzar el ciclo (Ver Código 5).

No olvidemos que, al margen de estos controles de estado, es necesario controlar en esta función la vida de la planta y si ha recibido el impacto de cualquier disparo (Ver Código 6).

Observamos cómo la vida depende del tipo de disparo que haya impactado, determinado por la variable "tipo_armamento".

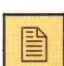
```
dreeck=LoadAnimMesh("dreeck.b3d")
ExtractAnimSeq(dreeck,0,30)
ExtractAnimSeq(dreeck,31,90)
ExtractAnimSeq(dreeck,91,140) ; 3
...
Animate dreeck,Modo,Vel,3,Trans
```

↓

```
dreeck=LoadMD2("dreeck.md2")
AnimateMD2 dreeck,Modo,Vel,91,140
```

6

Diferencia entre la utilización del formato .B3D al .MD2.



NOTA

Incluir los contadores y demás variables como los *switchers* en la estructura "tipo.dreeck" permite independizar la actividad de cada planta colocada en el terreno de juego.

Código 2. Estado de reposo de la planta

```
;reposo
If (dreeck\tiempo_cambio<dreeck\tiempo_cambio_total/4 And dreeck\sw_movimiento=1) And dreeck\ataque=0
   Animate dreeck\entidad_dreeck,1,Rnd(.3,.5),1,30
   dreeck\tiempo_cambio_total=dreeck\tiempo_cambio
   dreeck\sw_movimiento=2
EndIf
```

Código 3. Estado de acecho de la planta

```
If (dreeck\tiempo_cambio<dreeck\tiempo_cambio_total/2 And dreeck\sw_movimiento=0) And dreeck\ataque=0
   Animate dreeck\entidad_dreeck,1,Rnd(.3,.5),2,30
   TurnEntity dreeck\entidad_dreeck,0,3,0
   dreeck\sw_movimiento=1
EndIf
```

Código 4. La planta ataca si no hay camuflaje

```
;ataca si no hay camuflaje
If dreeck\distancia_comida#<200 And dreeck\ataque=0 And camuflaje_bionave=0
    Animate dreeck\entidad_dreeck,1,Rnd(.3,.5),3,30
    dreeck\ataque=1
EndIf
```

Código 5. Vuelve a comenzar el ciclo

```
If dreeck\tiempo_cambio=0
        dreeck\tiempo_cambio=Rnd(200,400)
        dreeck\tiempo_cambio_total=dreeck\tiempo_cambio
        dreeck\sw_movimiento=0
        dreeck\ataque=0
        Animate dreeck\entidad_dreeck,1,Rnd(.3,.5),1,30
EndIf
```

Código 6. Controlamos la vida de la planta

```
If dreeck\vida_dreeck<1 Then
   puntos=puntos+200
   FreeEntity dreeck\entidad_dreeck
   Delete dreeck
   Return
EndIf
If EntityCollided(dreeck\entidad_dreeck,ENTIDAD_DISPARO)
   dreeck\vida_dreeck=dreeck\vida_dreeck-(tipo_armamento*3)
   puntos=puntos+20
EndIf
```

Código 7. En los Lunys, sustituimos tiempo_cambio por tiempo_ataque

```
If luny\tiempo_ataque>0
    luny\tiempo_ataque=luny\tiempo_ataque-1
    TurnEntity luny\entidad_luny,0,1,0
   ;ataque
    If luny\tiempo_ataque<luny\tiempo_ataque_total/2 And luny\
    sw_movimiento=0
       Animate luny\entidad_luny,1,.8,2,30
       luny\sw_movimiento=1
    EndIf
   ;lanzar esporas
    If luny\tiempo_ataque<luny\tiempo_ataque_total/4 And luny\
    sw_movimiento=1
       emisorparticulas3( ......................
       Animate luny\entidad_luny,1,.3,1,10
       luny\sw_movimiento=2
    EndIf
EndIf
```

Código 8. Movimiento de los Slunks

```
If slunk\tiempo_cambio>0
   slunk\tiempo_cambio=slunk\tiempo_cambio-1
  ;sale, ataca y entra en la tierra
   If slunk\tiempo_cambio<slunk\tiempo_cambio_total/2 And slunk\
   sw_movimiento=0
   Animate slunk\entidad_slunk,3,Rnd(.3,.5),2,30
   slunk\sw_movimiento=1
   EndIf
EndIf
```

(■) COMPORTAMIENTO DE LOS DEMÁS SERES

El sistema utilizado para las plantas carnívoras es el que utilizamos para los Slunks y Lunys. La única diferencia son los tipos de estado de cada criatura. Como las variables de control de estado (contadores y switchers) pertenecen a la estructura de cada tipo de entidad, tendrán el mismo nombre y misión dentro de la función. Los Lunys no detectan la proximidad de la bionave, solo varían su estado según el control del contador "tiempo_cambio".

(Para los Lunys, la variable "tiempo_cambio" es sustituida por "tiempo_ataque") (Ver Código 7).

Los Slunks tampoco detectan la proximidad de la bionave, ya que su único estado en movimiento agrupa toda la activa en una sola animación. También viene determinado por el contador "tiempo_cambio" (Ver Código 8).

Sin embargo, añadir una detección de proximidad es fácil cambiando la sentencia condicional de este modo:

```
If (slunk\tiempo_cambio<slunk\
tiempo_cambio_total/2 And
slunk\sw_movimiento=0) Or
 EntityDistance (dreeck\
 entidad_slunk,bionave) < 200
```

Como hemos podido comprobar, implementar un comportamiento básico es realmente sencillo y todo depende de sentencias condicionales que detecten los cambios de la acción y, como siempre, de nuestra imaginación.

En el próximo número...

... seguiremos desarrollando la IA del juego aplicando el comportamiento de los OVNIS y los VOLADORES.

Fabricando el terreno de juego

En esta entrega vamos a aprender cómo realizar de varias formas el terreno para nuestro juego.

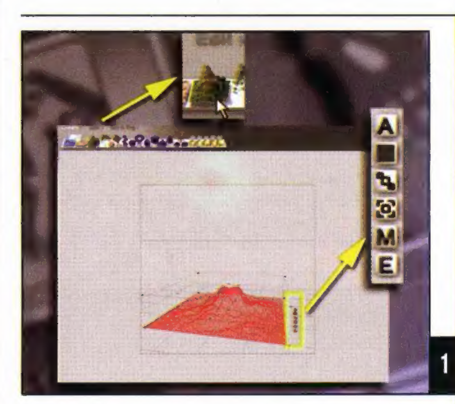
CONCEPTOS BÁSICOS

Básicamente, hay dos maneras de disponer de un terreno en Blitz3D: utilizando un *mesh* y mediante un mapa de alturas. La utilización de un modelo 3D (*mesh*) es la manera más natural de disponer de un terreno en nuestro juego. Este método tiene la ventaja de proporcionar un mayor control de todos los polígonos pero es más complicado y laborioso. Es muy utilizado para realizar, por ejemplo, circuitos para juegos de carreras de rally o simulaciones precisas de espacios abiertos. Con este sistema se puede crear un terreno a partir de un solo modelo o mediante la unión de varios; por ejemplo, podemos crear una cordillera uniendo diferentes modelos 3D de montañas. El segundo de los métodos (el mapa de alturas) es más sencillo de utilizar y consiste en representar alturas a partir de tonos de grises contenidos en un mapa de bits. Para crear el terreno en "Zone of Fighters" vamos a utilizar ambos métodos, los cuales desarrollaremos con la aplicación Bryce 5 y Paint Shop Pro. Sin embargo, aprovechando la cualidad de Blitz3D para generar un *mesh* a partir de un mapa de alturas, nos quedaremos al final con el segundo de ellos.

El programa Bryce es ideal para la realización de terrenos utilizando mallas a partir de un mapa de alturas. Sin embargo, solo permite exportar el *mesh* y las texturas que le apliquemos al terreno y no el mapa de alturas en sí. Pero nos ayuda a tener una referencia de cómo será nuestro terreno. Por otro lado, en Paint Shop Pro podemos dibujar nuestro mapa de alturas directamente e ir probando el resultado con Blitz3D o Bryce como veremos más adelante. Otra opción es ayudarnos de la imagen de las texturas que genera Bryce para dibujar en Paint Shop Pro por el método de capas superpuestas.

CREANDO NUESTRO TERRENO CON BRYCE. USANDO UN *MESH*

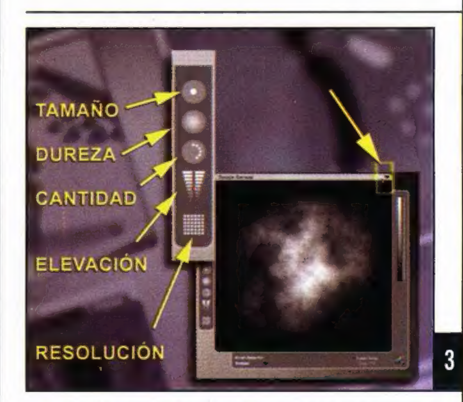
Ejecutemos la aplicación Bryce 5. Una vez dentro, lo primero que tenemos que hacer es crear un documento nuevo de 512 x 512 (*File/New Document* – "Ctrl" + "N") o modificar el existente por defecto (*File/Document Setup* – "Ctrl"+"Alt"+"N"). En la ventana *Document Setup* no olvidemos deseleccionar *Constrain Proportions*. La siguiente operación será crear la malla del terreno pulsando en el icono del menú *Create*. Bryce genera automáticamente un nuevo terreno con forma diferente cada vez que se pulsa en este icono. Entonces podemos observar cómo aparece una malla con la forma de una montaña en la vista. Para obtener una forma determinada de la malla debemos editarla. Para ello, pulsamos sobre la letra "E" que aparece en los iconos de



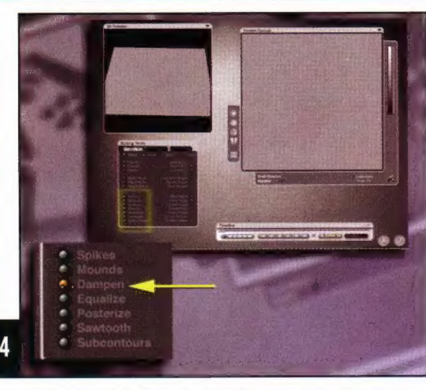
Procedimientos para crear un terreno y entrar en su editor.



En la ventana 3D Preview podemos ver la representación 3D del mapa de alturas desde todas direcciones.

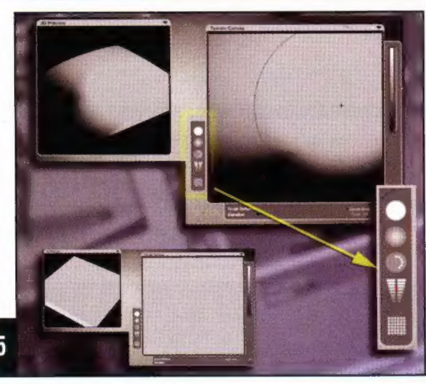


Descripción de las herramientas del pincel.



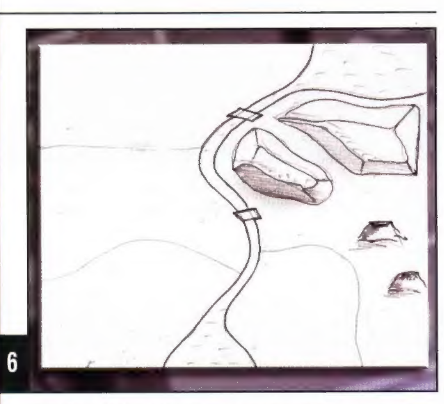
4

En la pestaña Elevation de la ventana Editing Tools encontramos todos los efectos que podemos aplicar al mapa de altura.



5

Antes de dibujar las diferentes alturas es necesario dibujar un terreno con una altura media uniforme.



6

Diseño original del terreno que tenemos que crear.

RECORDATORIO

Es importante recordar que, en una imagen para un mapa de alturas, las tonalidades negras representan las partes más bajas del terreno y las blancas las más altas.

opciones situados en la parte inferior derecha del terreno creado y entraremos en el editor de terrenos de Bryce. En él podemos distinguir varias ventanas (Ver Fig. 1).

VENTANA *3D PREVIEW*

Esta ventana nos muestra un previo de cómo será el terreno que estamos editando. Navegar por la vista es muy sencillo. Si queremos rotar el terreno, hacemos clic sobre la ventana y sin dejar de hacerlo movemos el ratón. Para acercarnos o alejarnos, movemos el ratón hacia delante o atrás, manteniendo pulsada la tecla "ALT". Para aumentar o disminuir la altura del terreno, movemos el ratón hacia delante o atrás mientras mantenemos pulsada la tecla "CTRL". Y para activar el "Auto Rotate", es decir, que el terreno gire sobre su eje Y, hacemos clic mientras mantenemos pulsada cualquier tecla "SHIFT". Pulsando sobre la flechita negra situada en la esquina superior derecha de la ventana entraremos en una serie de opciones en la que podemos cambiar la resolución de la ventana (*Set Preview Size*), activar el *Auto Rotate*, hacer un renderizado del previo, etc. (Ver Fig. 2).

VENTANA *TERRAIN CANVAS*

En esta ventana es donde vamos a dibujar nuestro mapa de alturas. Podemos observar cómo el terreno se representa como un dibujo de manchas blancas y negras. Realmente, es una zona de dibujo en la que podemos pintar con el ratón. A la izquierda de la ventana tenemos las opciones de pintado, las cuales describimos en la figura 3.

Pulsando en la flechita de color negro situada en la esquina superior derecha de la ventana entramos en el menú de opciones, en donde podemos modificar su tamaño. Al lado de esta flechita encontramos otro icono y al pulsarlo mostraremos algunas opciones más, entre ellas las del comportamiento del pincel.

VENTANA *EDITING TOOLS*

En esta ventana encontramos todos los efectos que le podemos aplicar al mapa de alturas, así como la asignación de imágenes o filtros. En nuestro terreno, solo vamos a utilizar efectos de elevación ("Elevation").

DIBUJANDO NUESTRO TERRENO

Vamos a comenzar realizando el mapa de alturas de nuestro terreno. Pero antes vamos a cambiar la resolución de la ventana de dibujo. Para ello, pulsamos en el icono de resolución y elegimos "512 - ultra-fine". El siguiente paso será rellenar toda la imagen con una altura media para, posteriormente, ir creando las partes altas y bajas del terreno. Para realizar esta operación podemos utilizar varias maneras. Una de ellas es aplicar un efecto *Dampen* (situado en la pestaña *Elevation* de la ventana *Editing Tools*) y la otra borrando el terreno que tenemos por defecto pulsando en el botón "New" situado en esta misma ventana. Para aplicar el efecto *Dampen*, nos situamos sobre la esfera verde situada junto al nombre "Dampen" y sin dejar de hacer clic desplazamos el ratón hacia el lado izquierdo hasta que veamos en la ventana *Terrain Canvas* que toda la imagen se vuelve de un tono uniforme gris. A partir de ahí, dibujamos las diferentes alturas (Ver Fig. 4).

Si por el contrario decidimos utilizar una imagen nueva, debemos pintar toda la ventana de dibujo con una altura uniforme. Para ello, colocamos el punto rojo de altura del icono en la mitad

de la escala. Seguidamente, seleccionamos la anchura máxima del pincel en el icono de tamaño, la máxima dureza (segundo icono) y un flujo de tinta máximo (tercer icono) (Ver Fig. 5).

Ya estamos preparados para definir todas las alturas del terreno de combate.

En primer lugar, vamos a dibujar el río que atraviesa el terreno según el diseño original (Ver Fig. 6) para dividir las dos partes del terreno. Elegimos un tamaño pequeño de pincel, la mínima dureza y un flujo medio de tinta. Y para finalizar, seleccionamos la altura mínima. Dibujamos con cuidado hasta obtener el resultado que podemos ver en la figura 7.

El siguiente paso será rodear el terreno de un muro. Así que tenemos que elegir la altura máxima (color blanco). Básicamente, ya tenemos la forma principal del terreno terminada. Nos queda dibujar los soportes para los puentes, las rampas de acceso desde el río y algunas montañas, entre ellas los volcanes. Recordaremos que la suavidad de bordes en el dibujo determinará unos cambios de altura redondeados. Para que el terreno no sea uniforme, podemos variar las alturas colocando algunas pinceladas de diferentes tonos. También sería interesante aplicar un efecto de erosión con la opción *Eroded* en la ventana *Editing Tools*. Hay que tener en cuenta que la escala del dibujo es muy pequeña con relación a la que necesitaremos en el juego. Por lo tanto, cualquier fisura o cambio de altura en el terreno se verá multiplicada en gran medida cuando estemos visualizándolo con Blitz3D. Básicamente, el trabajo que nos queda es conformar las diferentes partes del terreno, siempre siguiendo el diseño original. Por ejemplo, para el desfiladero, seleccionamos la altura máxima y pintamos. Luego, aplicamos un poco de erosión en los bordes cambiando el comportamiento del pincel en *Brush Behavior*. Para ello, pulsamos en la flechita negra situada junto a *Elevation*. Aparece un menú con los distintos comportamientos que puede adquirir el pincel y elegimos *Erosion*. La cantidad de erosión viene determinada por el tamaño del pincel. Para realizar los volcanes, solo tenemos que colocar un punto blanco (máxima altura) y otro más pequeño de color negro (mínima altura) en el centro para representar el cráter (no olvidemos volver a colocar el comportamiento del pincel en la opción *Elevation*). Antes de terminar no olvidemos dibujar los soportes para los puentes (Ver Fig. 8).

EXPORTANDO EL TERRENO

Una vez terminado de dibujar el terreno ya podemos exportar la malla que lo representa. Pero antes, es necesario aplicarle una textura. Volvemos a la vista principal pulsando en el icono de aceptar y entramos en el laboratorio de materiales. Por ejemplo, elegimos el material *Mid-Winter* de la librería *Planes & Terrains*. Una vez aplicado el material, volvemos a la zona de edición. Elegimos la opción *Export* situada en el menú (flechita negra) de la ventana *Terrain Canvas*. A continuación, el programa nos pedirá el nombre del archivo que contendrá el *mesh* del terreno. Seleccionamos el formato .3DS (versión 5) y un nombre. Al pulsar "Guardar" aparecerá la sección *Export Terrain*. Aquí, vemos una ventana en la que se muestra una representación 3D del terreno, que podemos rotar y mover como viene siendo habitual para todas las ventanas de previo. Vamos a modificar el número de polígonos del terreno a



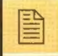
7

Aspecto que debe tomar el río que atraviesa la zona de combate.



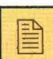
8

Aspecto definitivo del mapa de alturas.



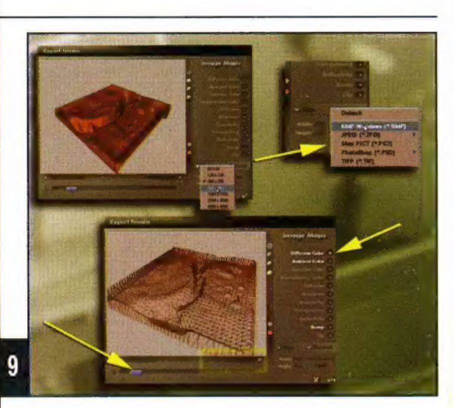
NOTA

En Bryce 5 todas las ventanas del modo edición de terrenos se pueden desplazar como las propias de Windows.



NOTA

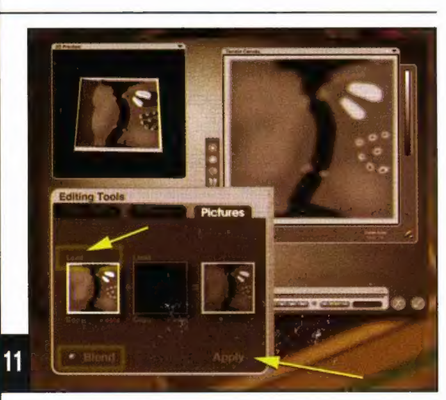
Para visualizar el resultado en tiempo real en la ventana *3D preview* mientras estamos pintando, debemos activar la opción *Realtime Linking* del menú de *opciones* de dicha ventana. También, en esta ventana podemos apreciar un renderizado del terreno activando la opción *Rendered Preview*.



Bryce nos permite exportar el terreno como un mesh y también las distintas propiedades del material.



También es posible crear con Paint Shop Pro el mapa de alturas utilizando el pincel.



Podemos utilizar Bryce para visualizar un mapa de alturas dibujado con Paint Shop Pro.

3200 desplazando la barra deslizadora azul hacia la izquierda. A la derecha de la ventana encontramos el apartado *Image Maps*, el cual muestra todas las cualidades del material que se exportará en imágenes; es decir, si tenemos activado *Diffusion Color* y *Bump* obtendremos dos imágenes, una para cada propiedad. En *Size* elegimos el tamaño de las imágenes y en *Forma* el formato. Por defecto tenemos un tamaño de 256 x 256, así que cambiémoslo a 512 x 512. y elijamos el formato .bmp. Como comentábamos, estas imágenes se pueden utilizar como referencia en Paint Shop Pro para desarrollar el mapa de alturas. Como solo necesitaremos la capa *Diffusion Color* para obtener esta referencia, deseleccionamos el resto de las opciones del material (Ver Fig. 9).

CREANDO EL MAPA DE ALTURAS CON PAINT SHOP PRO

Este método es más artesanal ya que tenemos que dibujar las diferentes alturas utilizando tonos de grises. Para empezar, creamos un documento nuevo de 512 x 512. Seguidamente, seleccionamos un tono de color gris y rellenamos la pantalla con la herramienta de relleno. Prácticamente, el mapa se dibujará de la misma manera que en Bryce, eligiendo tonos grises claros para las elevaciones y tonos oscuros para las hondonadas. Es importante jugar con los parámetros del pincel para obtener trazos suaves y así evitar brusquedad en los cambios de tonos. Como siempre, lo primero que vamos a dibujar es el río en el centro y los muros que delimitan el terreno. A continuación, los soportes para los puentes, el desfiladero y los volcanes. Terminaremos, haciendo menos uniforme el terreno con algunos cambios de tono para simular montículos y hondonadas (Ver Fig. 10).

Para comprobar el resultado podemos utilizar el visualizador de terrenos que desarrollamos en el número 7 del curso, o bien utilizar Bryce.

COMPROBANDO EL RESULTADO EN BRYCE

Grabemos el mapa de alturas dibujado en Paint Shop Pro en formato .bmp y ejecutemos Bryce. Dentro del programa, asignamos un tamaño al documento de 512 x 512. Seguidamente, creamos un terreno y entramos a editarlo. En la zona de edición elegimos la opción *New* en la pestaña *Elevation* de la ventana *Editting Tools*. Luego pulsamos en la pestaña *Pictures*. Aparecerán tres ventanitas separadas por los signos "+" y "=". Esto representa la posibilidad de utilizar cualquier imagen o varias mezcladas para crear un terreno. Hacemos clic en *Load* de la primera ventana y cargamos el mapa de alturas que dibujamos con Paint Shop Pro. A continuación, hacemos clic sobre la segunda ventana hasta que la imagen desaparezca. Seguidamente, desplazamos el ratón hacia la izquierda sobre *Blend* (mezcla), sin dejar de hacer clic, para modificar la mezcla entre las dos ventanas. El resultado (tercera ventana) tiene que ser igual a la primera ventana. Si pulsamos sobre la ventana del resultado (3ª ventana) se mostrará en *3D Preview* una representación del terreno. Al pulsar en *Apply* se usará nuestra imagen como terreno actual (Ver Fig. 11).



En el próximo número...

...desarrollaremos las texturas que cubrirán el terreno.

Nuestro primer tema musical con Anvil Studio (V)

Siguiendo con nuestro aprendizaje de Anvil Studio, continuaremos viendo las posibilidades del manejo de pistas de audio en nuestras canciones.

En el número anterior aprendimos la forma de grabar audio externo desde un micrófono directamente en una pista. Veremos a continuación cómo trabajar con una pista de audio y cómo obtener y editar ficheros de audio.

CARGANDO Y EDITANDO UN FICHERO DE AUDIO

Podemos importar directamente un fichero de audio en formato .WAV a una pista del secuenciador. La versión estándar de Anvil Studio solo puede mezclar una pista de audio a la vez, para obtener más pistas es necesario adquirir el accesorio "MultiAudio". Sin embargo, es suficiente para nuestros propósitos. Cambiamos la pista por defecto de *Instrument* a *Audio* en el campo *Type* y elegimos *Stereo* en el campo *Channel*.

Activamos la pista y ya estamos preparados para recibir un fichero externo de audio. Seleccionamos la opción *Import Audio to Active Track from... / Import entire file* del menú *File*. Si no hemos salvado la canción actual nos pedirá un nombre de canción para guardarla y luego podremos elegir el fichero de audio. Importamos "loop.wav" contenido en "Extras" del CD (Ver Fig. 1).

Veremos más adelante cómo podemos utilizar ficheros de audio como sonidos para nuestras pistas de ritmos. Una vez cargado, podemos ver y editar la forma de onda pulsando en el botón "Compose". Para reproducir el audio pulsamos en el botón junto a "Play this track". Podemos también reproducir solo una parte del audio seleccionándola con el ratón (Ver Fig. 2).

A la izquierda de la ventana que muestra la forma de onda encontramos 5 botones: "Louder", "Volume", "Pitch", "Filter" y "Mixer". El primero de ellos ("Louder") sirve para doblar el volumen actual de la muestra (hay que tener en cuenta que este tipo de operaciones puede ocasionar distorsión en el audio). En "Volume" podemos modificar el volumen de la muestra (desde 0 hasta 3) de forma controlada a través de un deslizador. "Pitch" sirve para cambiar el tono de la onda. El botón "Filter" nos lleva a una librería de filtros para aplicar a la muestra. Entre ellos encontramos el normalizado, que como sabemos, es muy útil para subir al máximo el volumen de la muestra sin distorsionar. Con el último botón ("Mixer") salimos de *Compose* (Ver Fig. 3).

El deslizador "Zoom" que hallamos en la parte inferior nos ayuda a aumentar la muestra en el tiempo y el deslizador vertical situado justo al final de la ventana de visualizado de onda, sirve para aumentar la muestra en la gráfica del volumen. Otro efecto que podemos aplicar es dar la vuelta a la muestra o parte de ella con la opción *Reverse* del menú *Edit*. Par poder aplicarlo es necesario seleccionar parte de la muestra con el ratón o seleccionándola por completo con *Edit/Select All* ("Ctrl" + "A").

También es posible realizar operaciones de edición como



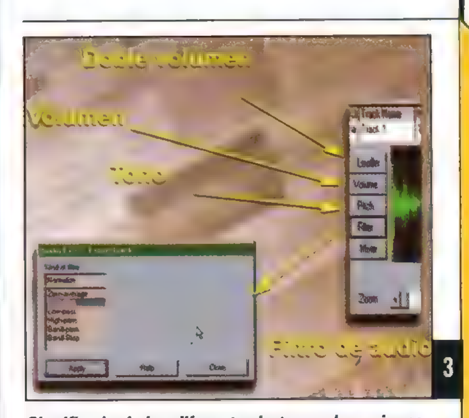
1

Podemos importar un fichero de audio en formato .WAV directamente a la pista.



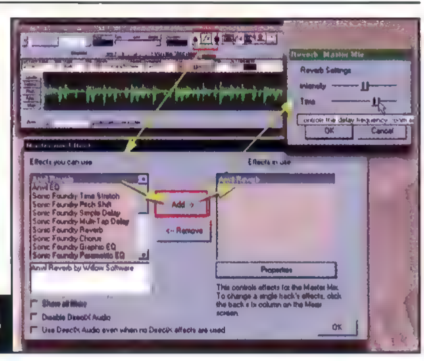
2

AL pulsar en el botón "Compose" entramos en la ventana de edición de la muestra.

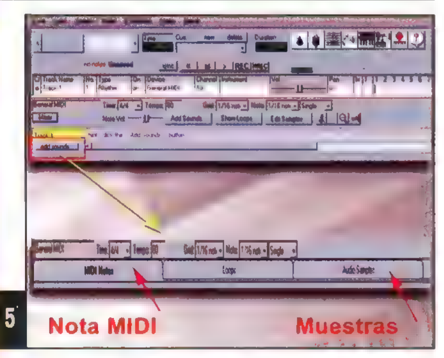


3

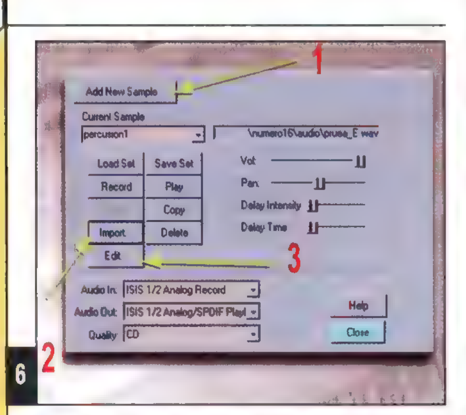
Significado de los diferentes botones de opciones para modificar la muestra de audio.



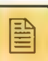
Además de los efectos propios de Anvil Studio, podemos añadir todos los efectos compatibles con DirectX instalados en el sistema.



Las pistas de ritmos son ideales para secuenciar partes con muestras y notas MIDI.



Pasos para añadir un nuevo sonido de muestra.



NOTA

Al cambiar la pista activa en pista de audio, podemos observar cómo en los iconos situados en la parte superior derecha aparece un icono nuevo, el cual sirve para acceder a los efectos de audio.

cortar, borrar, pegar o insertar con partes de la muestra. Para ello, solo tenemos que seleccionarla y elegir las opciones. Las operaciones de edición funcionan exactamente igual que en otros tipos de modos y pistas.

APLICANDO EFECTOS A LA PISTA DE AUDIO

Es posible aplicar otros tipos de efectos a la muestra completa o parte de ella. Solo tenemos que pulsar en el icono y entraremos en una ventana flotante con las librerías disponibles. Anvil Studio cargará automáticamente todos los efectos compatibles con DirectX que tengamos instalados en nuestro sistema además, claro está, de los efectos propios del programa. Para aplicar el efecto, debemos seleccionarlo de la lista *Effects you can use* y pulsar en el botón "Add —>" para pasarlo a la lista *Effects in use*. Para modificar los parámetros de los efectos, solo tenemos que pulsar en el botón "Properties" y se abrirá otra ventana flotante con sus cualidades (Ver Fig. 4).

CREANDO PISTAS DE RITMOS

El último tipo de pistas que podemos crear en Anvil Studio son las de ritmos. Estas pistas se pueden considerar como una mezcla entre "Instrument" y "Audio" ya que permiten mezclar notas MIDI y muestras. Creamos una canción nueva. En la pista por defecto cambiamos el tipo (Type) a pista de ritmo (Rhythm). Para editarla, pulsamos en el botón "Compose". Para poder añadir notas a la pista es necesario seleccionar antes un sonido. Para ello, pulsamos en el botón "add sounds..." situado en la parte inferior. Aparecerá la lista de notas MIDI en la pestaña "MIDI notes". Pero también podemos seleccionar loops o muestras, en las pestañas "Loops" y "Audio Samples" respectivamente (Ver Fig. 5).

Vamos a elegir un sonido perteneciente a una muestra. Pulsamos sobre la pestaña *Audio Samples* y después sobre *Edit Samples*. Aparece una nueva ventana en donde obtendremos la muestra importándola de disco o grabando directamente. En nuestro caso, vamos a cargar el sonido "HandClap.wav". Antes de poder importar debemos asignar mediante un nombre (por ejemplo "percusion1") la nueva muestra al programa pulsando sobre el botón "Add New Sample". Después de esto ya podemos cargar la muestra pulsando sobre el botón "Import...". Una vez cargada, podemos editarla (botón "Edit...") o reproducirla (botón "Play") antes de cerrar o también modificar directamente su volumen, panorámico, etc. mediante los deslizadores Vol, Pan, etc. (Ver Fig. 6).

Al pulsar "Close" volvemos a la pantalla *Audio Samples* en donde podemos apreciar que nuestra nueva muestra ha inaugurado la lista. Al volver a la cuadrícula vemos cómo debajo de "Track1" (en amarillo) aparece un botón con el nombre de la nueva muestra. Si pulsamos sobre él sonará. Ya solo tenemos que añadir sobre la cuadrícula las notas, haciendo clic con el ratón. Podemos sumar a la lista cuantos sonidos de muestras queramos, además, claro está, de sonidos MIDI.

Este sistema para añadir sonidos de muestras es también viable en pistas de tipo "Instrument"; la única diferencia es la forma de edición de este tipo de pistas, pero para todo lo demás es exactamente igual.



En el próximo número...

... terminaremos nuestro aprendizaje de Anvil Studio viendo el funcionamiento de la opción "Performer" y de cómo realizar Loops.

Manejo de ficheros y los bancos de memoria

Después de tratar en el número anterior la gestión del teclado, ratón y dispositivos de juego, en esta ocasión aprenderemos a utilizar los ficheros y todos los procesos derivados de su uso.

Debemos distinguir entre la gestión del fichero en sí mismo y el manejo de los datos que contiene. Por lo tanto, aprenderemos a crear, borrar, buscar, etc. ficheros en el disco, así como a trabajar con los directorios. Y por otro lado, veremos la manera de grabar o leer datos de un fichero.

Como sabemos, el uso de ficheros es fundamental en cualquier aplicación. Siempre es interesante poder grabar datos de utilización interna de un programa en un soporte físico para uso posterior o para intercambio con otras aplicaciones. Desde un sencillo archivo de configuración hasta una gran base de datos, todo es posible en Blitz3D. Además, se dispone de instrucciones para el acceso secuencial y aleatorio de los datos contenidos en un fichero, este último a través de punteros de localización realmente flexibles.

```
Dir= ReadDir("c:\")
Repeat
  F$=NextFile(Dir)
  Print F$
Until F$=""
```

1

Código para mostrar los ficheros contenidos en un directorio.

MANEJANDO DIRECTORIOS

Antes de trabajar con los ficheros individualmente, vamos a conocer cómo controlamos el uso de los directorios donde ubicaremos nuestros archivos. Empezaremos, conociendo cuál es el directorio actual seleccionado, en el que se situarán todas las operaciones con archivos. Para ello tenemos la instrucción "CurrentDir$()". Esta instrucción devuelve un *string* (lista de caracteres alfanuméricos) con el nombre del directorio actual en uso:

```
              = CurrentDir$()
Print
```

Si queremos cambiarlo, disponemos de "ChangeDir":

```
ChangeDir "  :  .              "
```

Aquí, cambiamos el directorio actual a "datos".

Crear o borrar un directorio es fácil con "CreateDir" y "DeleteDir" respectivamente:

```
              = "  .              "
CreateDir
DeleteDir
```

Otra función interesante que podemos usar es "ReadDir", la cual permite leer el contenido de un directorio; es decir, todos los archivos y carpetas que contiene. Sin embargo, para sacar todo el provecho a esta instrucción debemos utilizarla en combinación con "NextFile$" o "FileType". No olvidemos que después de abrir un directorio y hacer operaciones con él es necesario cerrarlo con "CloseDir".

```
CurrentDir
ChangeDir
CreateDir
DeleteDir
ReadDir
CloseDir
```

2

Lista de comnados disponibles para el manejo de directorios.

OPERACIONES CON LOS ARCHIVOS DE UN DIRECTORIO

Como comentábamos, "ReadDir" solo lee el directorio, así que para leer cada archivo o carpeta del mismo debemos recorrerlo con la instrucción "NextFile$":

```
   =ReadDir("c:\")
Repeat
    $=NextFile(Dir)
   Print F$
Until F$=""
```

"NextFile$" proporciona el nombre y la extensión del siguiente fichero del directorio leído por "ReadDir" pero no

NOTA

No podemos recorrer un directorio hacia atrás, solo hacia delante. Para poder trabajar con el directorio a nuestro antojo debemos introducir su contenido en una matriz. De esta forma, podemos, por ejemplo, ordenarlo o recorrerlo en cualquier sentido.

OpenFile
ReadFile
WriteFile
CloseFile
CopyFile
DeleteFile

3

Lista de comandos básicos para el manejo de ficheros.

sabe distinguir si se trata de un archivo o de una carpeta, para ello debemos utilizar la instrucción "FileType".

```
FileType ([illegible] $)
```

Este comando devuelve 3 valores:

- **1.** El nombre del fichero existe.
- **0.** El nombre del fichero no existe.
- **2.** El nombre del fichero corresponde a un directorio.

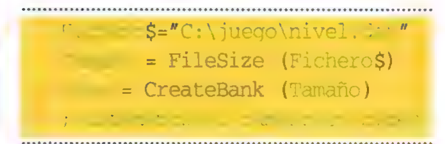
```
[illegible] $="C:\juego\config.dat"
If FileType([illegible] $)=0 Print
"[illegible]
[illegible]"
```

NextFile
FilePos
SeekFile
FileType
FileSize
ExecFile

4

Lista de comandos especiales para el manejo de los datos de un fichero.

A menudo, necesitaremos saber el tamaño en bytes de un fichero, bien para copiarlo, leerlo o almacenarlo en un banco de memoria. Para disponer de esta información tenemos la instrucción "FileSize":



```
[illegible]$="C:\juego\nivel.[illegible]"
[illegible] = FileSize (Fichero$)
[illegible] = CreateBank (Tamaño)
;[illegible]
```

Podemos también actuar directamente con los ficheros de un directorio determinado. Así, si queremos borrar un fichero, podemos hacerlo con "DeleteFile":

```
DeleteFile "C:\juego\datos.dat"
```

Es necesario asegurarse de que el fichero que vamos a borrar existe. Si no existe, el programa no dará error pero, evidentemente, no habrá ninguna consecuencia en el directorio.

Una opción muy útil: si queremos construir nuestro propio instalador, está la necesidad de copiar ficheros de una ubicación a otra. Para ello, Blitz3D nos proporciona la instrucción "CopyFile":

```
CopyFile ubicación origen$,
[illegible]estino$
CopyFile "C:\pantalla.jpg"
"[illegible]"
```

Y por último, una instrucción que nos permitirá ejecutar un programa ejecutable externo, "ExecFile". Muy interesante, por ejemplo, para visualizar un fichero de texto desde Blitz3D, como la ayuda del juego, etc. (Ver "ejemplo1.bb"):

```
ExecFile ("[illegible]
[illegible]")
```

MANEJANDO ARCHIVOS Y EDITANDO SU CONTENIDO

A continuación, vamos a aprender cómo crear y utilizar nuestros propios ficheros para grabar o leer datos. Antes de cualquier operación de lectura o escritura en un fichero nuevo, debemos crearlo. Para ello, utilizamos la instrucción "WriteFile". En realidad, lo que este comando hace es abrir un fichero y lo prepara para que se puedan grabar datos en él. Si el fichero no existe, lo crea:

```
[illegible] = WriteFile
("[illegible]")
```

Si ya tenemos un fichero creado con datos y queremos leer de él, debemos utilizar "ReadFile" para abrirlo:



Esquema de la disposición de números enteros en un fichero.

NOTA

Si abrimos un fichero con "WriteFile" y el fichero ya existe, inicializará el archivo borrando todos los datos aunque no hagamos ninguna operación en él.

RECORDATORIO

Vamos a recordar los valores que toman los diferentes tipos de datos en Blitz3D:

Byte	**0 - 255**
Short	**0 - 65535**
Int	**-2147483647 - 2147483647**
Float	**-3.4 x 10 ^ -38 - 3.4 x 10 ^ -38 .**



Cálculo del desplazamiento para la búsqueda de registros en un fichero.

```
ReadFile (nombreFichero$).
```

Una vez que hayamos abierto un archivo y terminado las operaciones con él, es importante cerrarlo con la instrucción "CloseFile":

```
 = ReadFile ("Configuración.dat")
CloseFile
```

Blitz3D nos permite leer o grabar un dato determinado de un fichero sin tener que realizar ninguna copia del fichero o trabajar con todos los registros secuencialmente. Disponemos para ello de "FilePos" y "SeekFile".

El primero de ellos, "FilePos", devuelve la posición actual que esta siendo editada (leída, escrita o modificada) dentro del fichero.

```
Fichero = WriteFile("datos.dat")
For n = 1 To 8
  Numero=Rand (1,100)
  WriteInt ( Fichero, Numero )
Next
CloseFile (Fichero)
; -----------------------------
Fichero = ReadFile("datos.dat" )
For n=1 To 8
  Numero = ReadInt(Fichero)
  Posicion = FilePos (Fichero) - 4
  Print "Numero: " + Numero +
  " Posición "+ Posicion
Next
CloseFile( Fichero )
```

En este ejemplo, creamos un fichero ("datos.dat") y grabamos en él 8 números enteros aleatorios. Seguidamente, volvemos a abrirlo para leer los 8 números. En las líneas:

```
Posicion = FilePos (Fichero) - 4
Print "Numero: " + Numero +
" Posición "+ Posicion
```

Imprimimos en pantalla el número contenido en el fichero y su posición. Debemos observar cómo restamos 4 a la posición, ya que la primera corresponde a 0.

Sin embargo, podemos también desplazarnos por las posiciones de un archivo para poder editar sus datos individualmente. Para ello, disponemos de la instrucción "SeekFile":

```
SeekFile
```

Vamos a añadir al ejemplo anterior las siguientes líneas:

```
SeekFile (Fichero,4)
Print "El numero de la posición
4 es: " + ReadInt(Fichero)
CloseFile(Fichero)
```

Aquí, nos situamos en la segunda posición, correspondiente a un desplazamiento de 4 bytes (números enteros), y leemos el número grabado en dicha posición (Ver "ejemplo2.bb").

Generalmente, no sabemos el número de datos que compone un fichero y necesitamos leerlo en su totalidad. Para ello, es necesario entrar en un bucle y leer todos los datos uno a uno. Para saber si hemos llegado al final del fichero disponemos de la instrucción "Eof". Supongamos que queremos imprimir todos los números enteros de un fichero:

```
 = ReadFile ("Datos.dat")
While Not Eof ( Fichero )
    Print ReadInt (Fichero)
Wend
```

Como hemos podido comprobar, manipular los datos de un fichero es realmente sencillo. Vamos a ver a continuación la forma de leer o escribir esos datos en un archivo según al tipo a que pertenecen.

Para cada tipo de datos disponemos de una instrucción específica de lectura y escritura.

LEER Y ESCRIBIR DATOS NUMÉRICOS

Como sabemos, disponemos de cuatro tipos de datos numéricos en Blitz3D: Byte, Short, Int y Float.

En la página siguiente mostramos una tabla con las diferentes instrucciones para leer o escribir estos tipos de datos (Ver Cuadro 1).

LEER Y ESCRIBIR DATOS ALFANUMÉRICOS

Podemos leer o escribir una variable alfanumérica (string) o una línea completa de texto. Esto último es ideal para trabajar con documentos de texto (Ver Cuadro 2).

RELACIÓN ENTRE LOS FICHEROS Y LOS BANCOS DE MEMORIA

La utilización de bancos de memoria es ideal para realizar



NOTA

Las posiciones dentro de un fichero corresponden a los bytes que ocupan los datos contenidos en él. Por ejemplo, un número entero tiene 4 bytes, así que ocupará 4 posiciones en el fichero. Si tuviéramos 2 números enteros, el primero comenzaría en la posición 0 y el segundo en la 4.

TRUCO

Para calcular el desplazamiento necesario para averiguar la posición de un elemento dentro de un fichero podemos utilizar la siguiente fórmula:
Desplazamiento = Elemento * Tamaño del elemento – Tamaño del elemento.

procesos de empaquetamiento o encriptación de datos debido a la gran velocidad de transferencia de datos que permite. Blitz3D nos proporciona dos funciones que permiten leer o escribir directamente de un fichero a un banco de memoria: "ReadBytes" y "WriteBytes".

"WriteBytes" permite escribir datos de un banco de memoria a un fichero. El fichero debe ser abierto con "WriteFile". Después de las operaciones de escritura debe ser cerrado con "CloseFile".

```
WriteBytes [illegible]
```

En el siguiente ejemplo, creamos un banco de memoria para llenarlo de bytes y números enteros (10 de cada uno) Posteriormente, abrimos un fichero para introducir estos números en él con "WriteBytes". Para finalizar, cerramos el fichero y borramos el banco de memoria:

```
Banco=CreateBank(100)
For Offset = 1 To 10
 PokeByte  Banco,Offset,Rnd(100)
 PokeInt Banco, Offset + 1,
 Rnd(-10,10)
Next
; Abrimos el fichero
Fichero=WriteFile ("Fichero.bnk")
WriteBytes Banco, Fichero, 0, 10
CloseFile Fichero
FreeBank Banco
```

Una vez que tenemos los números del banco de memoria en un fichero, podemos leerlos posteriormente e introducirlos en otro banco:

```
Banco=CreateBank(100)
; Abrimos el fichero
Fichero=OpenFile ("Fichero.bnk")
ReadBytes Banco, Fichero, 0, 10
CloseFile Fichero
For Offset = 1 To 10
 PeekByte  Banco, Offset
 PeekInt   Banco, Offset + 1
Next
```

Cuadro 1. Leer y escribir datos numéricos

Datos numéricos	Leer	Escribir
Byte (0 – 255)	**ReadByte** (Fichero)	**WriteByte** (Fichero)
Short (0-65535)	**ReadShort** (Fichero)	**WriteShort** (Fichero)
Int (-2147483648 – 2147483648)	**ReadInt** (Fichero)	**WriteInt** (Fichero)
Float (-3.4 x 10 ^ -38 - 3.4 x 10 ^ -38)	**ReadFloat** (Fichero)	**WriteFloat** (Fichero)

Cuadro 2. Leer y escribir datos alfanuméricos

Datos alfanuméricos	Leer	Escribir
String (variable alfanumérica)	**ReadString** (Fichero)	**WriteString** (Fichero)
Line (línea de texto)	**ReadLine** (Fichero)	**WriteLine** (Fichero)

Código 1. Grabando posiciones

```
; GRABACIÓN
If Offset>Memoria-8
  SW_Grab=False
  Fichero_Grabacion=WriteFile("Camara.dat")
  WriteBytes BankMemX,Fichero_Grabacion,0,Memoria
  WriteBytes BankMemY,Fichero_Grabacion,0,Memoria
  WriteBytes BankMemZ,Fichero_Grabacion,0,Memoria
  WriteBytes BankMemPitch,Fichero_Grabacion,0,Memoria
  WriteBytes BankMemYaw,Fichero_Grabacion,0,Memoria
  CloseFile Fichero_Grabacion
EndIf
```

Código 2. Reproducir el fichero

```
;Activar Reproducción
If KeyDown(57) And SW_Play=False
  SW_Play=True
  Offset=0
EndIf
If KeyDown(56) And SW_Play=False
  SW_Play=True
  Offset=0
  Fichero_Grabacion=OpenFile("Camara.dat")
  ReadBytes BankMemX,Fichero_Grabacion,0,Memoria
  ReadBytes BankMemY,Fichero_Grabacion,0,Memoria
  ReadBytes BankMemZ,Fichero_Grabacion,0,Memoria
  ReadBytes BankMemPitch,Fichero_Grabacion,0,Memoria
  ReadBytes BankMemYaw,Fichero_Grabacion,0,Memoria
  CloseFile Fichero_Grabacion
EndIf
```

Estos procedimientos se pueden utilizar en el sistema de grabación de posición de la cámara que desarrollamos en el tutorial del número 15. Podríamos añadir las siguientes líneas de código cuando estamos grabando posiciones y terminamos (Ver Código 1).

De esta forma hemos creado un fichero ("Camara.dat") con todas las posiciones contenidas en los bancos de memoria. Ahora solo queda añadir el código para leer el fichero y reproducirlo. Por ejemplo, al pulsar la tecla "Alt" (Ver Código 2).

En el próximo número...

... aprenderemos a utilizar el audio en nuestros programas y a manipular archivos de vídeo.

Realización de un vídeo para nuestro **juego** (I)

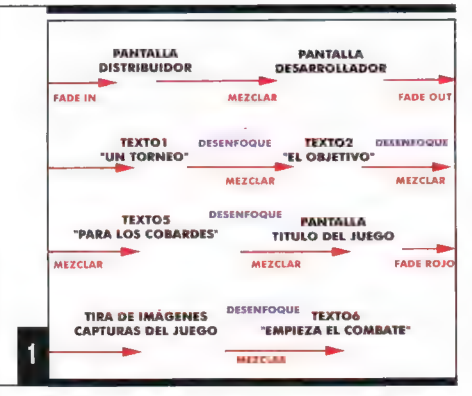
Con esta entrega vamos a desarrollar un pequeño vídeo que servirá para presentar nuestro juego.

Para ello utilizaremos el editor de vídeo de Ulead MediaStudio Pro 6.5 y Paint Shop Pro.

LA IDEA

Se puede conseguir una presentación digna con poco esfuerzo, solo es cuestión de tener las ideas claras de lo que se quiere hacer. Para nuestra presentación utilizaremos los logotipos del desarrollador, distribuidor y del juego. También utilizaremos algunas pantallas de texto y capturas del juego. La idea es mostrar poco a poco en pantalla algunas frases clave que desvelen el argumento del juego y que den paso a una secuencia de capturas. Antes de empezar es importante realizar un guión del vídeo para saber en todo momento cómo se desarrollará la presentación (Ver Fig. 1).

Para desarrollar las pantallas de texto vamos a utilizar Paint Shop Pro.



Guión básico del vídeo de presentación, gracias al cual sabremos en todo momento cómo se desarrollará.

DIBUJANDO LAS PANTALLAS DE TEXTO

Dentro del programa creamos un documento nuevo con fondo negro y de 800 x 600 de resolución. Seguidamente, elegimos la herramienta de texto para escribir la primera de las frases que aparecerá en pantalla. En la ventana de texto *Text Entry* escribimos la frase "UN TORNEO" con el tipo de letra que diseñamos para "Zone of Fighters" ("zof.ttf", "dungeon.ttf" modificada). Si no disponemos de la fuente se puede encontrar en el directorio "Extras" del CD.

Seleccionamos un tamaño de letra de 48 y un color de relleno *Styles / Fill* blanco. Los "Strokes" de *Styles* y *Textures* los anulamos, así como el "Fill" de *Textures* (Ver. Fig. 2).

Al aceptar hemos creado una nueva capa vectorial, la cual debemos rasterizar (convertir a mapa de bits) para poder aplicarle efectos de imagen. Una vez rasterizada le aplicaremos un efecto de textura para cambiar el aspecto de las letras. En el menú *Effects* elegimos la opción *Sculpture* en *Texture Effects*. Y aplicamos los siguientes parámetros:

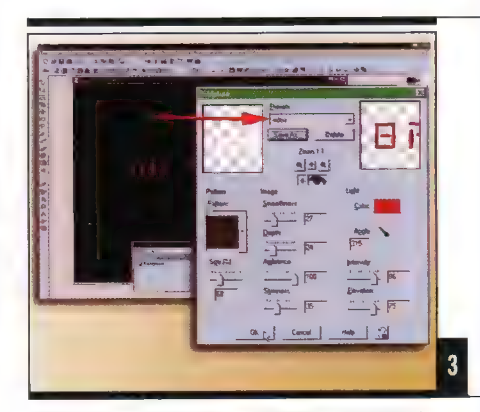
En *Pattern* seleccionamos el patrón número 21 (Leather.bmp) con un tamaño del 68%. En *Image* aplicamos un *Smoothness* de 27, un *Depth* de 24, un *Ambience* de 100 y un *Shininess* de 35. En la sección *Light* seleccionamos un ángulo *Angle* de 315, una *Intensity* del 86% y una *Elevation* de 75. Pulsamos en *Save As* ... para



Para escribir el texto de las frases utilizaremos la fuente del juego plana y de color blanco.

grabar nuestro efecto y poder así utilizarlo para los demás textos. Asignemos el nombre "vídeo" al nuevo *preset* (Ver Fig. 3).

Para el resto del texto es solo cuestión de escribirlos y aplicarles el nuevo efecto que hemos creado. Creamos una nueva capa para cada frase. Luego debemos grabar una imagen con cada una de ellas. Para ello, ocultamos las capas que no nos interesan y colocamos la frase, con la herramienta de transformación, más o menos en el centro



Un efecto de textura tipo "Sculpture" es ideal para obtener la apariencia deseada para las letras.



4

Una vez que tengamos todas las capas, grabamos una pantalla con cada frase. Para ello, mantenemos visible la que nos interesa.



5

Antes de comenzar a editar con Ulead MediaStudio es necesario elegir una plantilla para el proyecto.

de la pantalla. Salvamos en formato .bmp en *File/ Save As*. Seguiremos una secuencia de nombres, por ejemplo, "Texto1.bmp" para la frase "UN TORNEO", "Texto2.bmp" para "EL OBJETIVO" y así sucesivamente. En total debemos tener 6 pantallas (Ver Fig. 4):

"Texto1.bmp" ➡ "UN TORNEO"
"Texto2.bmp" ➡ "EL OBJETIVO"
"Texto3.bmp" ➡ "SOBREVIVIR"
"Texto4.bmp" ➡ "NO HAY PIEDAD"
"Texto5.bmp" ➡ "PARA LOS COBARDES"
"Texto6.bmp" ➡ "EMPIEZA EL COMBATE"

OBTENIENDO LAS CAPTURAS DEL JUEGO

La siguiente fase de nuestro vídeo es adquirir, desde el juego, las capturas que nos servirán para terminar nuestra presentación. Para ello, solamente tenemos que pulsar, durante una partida, la tecla "S". En ese instante el búfer de pantalla se grabará en un fichero de mapa de bits (.bmp) en el directorio del juego ("C: \ zone of fighters\"). Simplemente, se ha añadido al código del juego la siguiente función para poder realizar las capturas, la cual es llamada continuamente desde el bucle principal de la partida (Ver Código 1)

EL EDITOR DE VÍDEO

Una vez que tenemos todo el material gráfico pasamos al editor para conformar el vídeo. Hemos elegido Ulead MediaStudio porque posee un editor de vídeo muy sencillo de utilizar y a la vez potente. Suficiente para nuestras necesidades. No vamos a profundizar en todas las posibilidades que nos ofrece, pero sí aprenderemos lo fundamental para el desarrollo de cualquier presentación que se nos ocurra (Ver Fig. 5).

Al ejecutar la aplicación y antes de entrar en el editor, debemos seleccionar qué tipo de vídeo vamos a crear. Para ello, seleccionamos, de momento, la plantilla *Presentation Video* (25 fps, Indeo) de la lista de plantillas *Existing project templates*. Ya dentro del editor se nos presentan cuatro ventanas: *Timeline, Preview, Source* y *Production Library*.



6

Ventana principal del editor de Ulead MediaStudio Pro 6.5, con sus cuatro áreas principales: previo, fichero, efectos y el montaje propiamiente dicho.

En *TimeLine* es donde editamos nuestro vídeo, en *Preview* se muestra un previo de la edición, en *Source* se visualiza el resultado final y *Production Library* es un lista de todos los efectos disponibles para la producción audiovisual (Ver Fig. 6).

La versión *trial* de este programa mostrará continuamente en la pantalla de previo y en el vídeo una "X" de color rojo, aun así podemos ver cómo resulta nuestra pequeña producción.

Código 1. Función para obtener las capturas

```
Function Salvapantalla()
  ; salva pantallas con "S"
  If KeyDown (31)
    While fileExists("c:\zone of fighters\"+captura+".bmp")
      captura=captura+1
    Wend
    SaveBuffer (FrontBuffer(),"c:\zone of fighters\"+captura+".bmp")
    captura=captura+1
  EndIf
End Function
```

En el próximo número...

... entraremos de lleno en la edición del vídeo de presentación de nuestro juego.

Simuladores de vuelo

En esta entrega empezamos a recorrer la historia del género de la simulación con los simuladores de vuelo para PC.

Los simuladores de vuelo han estado siempre ligados al sector civil y militar de la aviación por los grandes beneficios que reportan en la detección de errores o en cursos de entrenamiento para pilotos. Al principio, solo los grandes ordenadores podían simular el comportamiento de un aeroplano con más o menos calidad. La llegada a los hogares de los microordenadores supuso, una vez más, un revulsivo para el desarrollo de simples simuladores para uso lúdico. La producción de juegos para ordenador basados en este género llevó al desarrollo de simuladores de muy diversa índole, buscando, cada vez más, la acción propia de un juego que la estricta aplicación de la simulación. Así, podemos encontrar simuladores aeronáuticos de aviones comerciales, de combate o de helicópteros. También, traspasando la frontera de lo real, el género se traslada a la ciencia-ficción, apareciendo simuladores espaciales.

SIMULADORES DE VUELO CIVILES

Básicamente, este tipo de simuladores ha creado el origen de este género para ordenadores personales. El concepto primordial es simular el comportamiento de aviones comerciales a través de gráficas 3D y control gráfico de instrumentales de navegación representado en pantalla. El primer juego basado en esta idea fue *Flight Simulator*. Fue desarrollado por Bruce ArtWick en 1979 para ordenadores Apple II después de fundar, con *Stu Moment*, la compañía SubLogic. Estaba basado en su tesis para la Universidad sobre la representación gráfica en tiempo real de una simulación de vuelo. Poseía gráficos en 4 colores y un panel de control con dos instrumentos de medida. No fue hasta 1982 cuando se desarrolla la versión para IBM PC. Editada por Microsoft, poseía 4 colores, gráficos vectoriales en 3D y hasta 8 instrumentos de medida. Además incorporaba climatología, 9 vistas diferentes y 4 escenarios con 20 aeropuertos. Todo un lujo en aquella época. El juego se convirtió en todo un éxito, hasta el punto que sus usuarios compraban mejores ordenadores solo para jugarlo. Durante los años 1984 y 1987 se realizaron 14 versiones para diferentes sistemas. Con la versión *Microsoft Flight Simulator 98* se incluyó la posibilidad de manejar helicópteros. El juego cumplía por entonces 15 años de existencia y ya había vendido la friolera de 10 millones de copias. Actualmente, *Flight Simulator* va por la octava generación con *Flight Simulator 2002* versión 8.0 y está en el libro Guinness de los récords al juego más vendido de la Historia con más de 25 millones de copias. Hoy día, *Flight Simulator* es un juego de culto en su género e insuperable y es utilizado en el sector aeronáutico como simulador de entrenamiento de pilotos. Al margen de *Flight Simulator*, otra desarrolladora compite con Microsoft en los simuladores civiles con la serie *Flight Unlimited*, nos referimos a Looking Glass Studios. *Flight Unlimited* se caracteriza por poseer gráficos de escenarios más realistas, con efectos especiales increíbles y una forma más entretenida de juego. Con *Flight Unlimited III* todos los escenarios están generados a partir de datos de satélites.



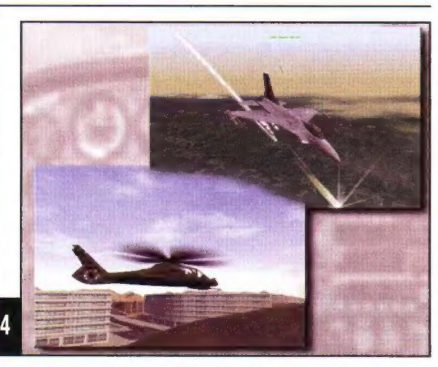
1 *Flight Simulator, el primer simulador de vuelo y el más vendido de la historia.*



2 *F15 Strike Eagle (arriba) y GunShip (abajo), entre los precursores de los actuales simuladores de vuelo de combate.*



3 *European Air War (arriba): máximo exponente de los simuladores de la Segunda Guerra Mundial. F/A-18 Simulator de Jane´s Combat Simulations.*



4

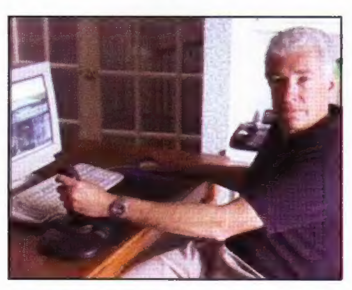
Falcon 4.0 (arriba) y Comanche 4 (abajo) forman parte de la serie de simuladores de vuelo de combate más populares.

LA BIOGRAFÍA...

BRUCE ARTWICK

Creador de **Flight Simulator**

Entró en la universidad para estudiar ingeniería de computadoras en 1973 pero cambió a ingeniería eléctrica; aun así, siempre estuvo interesado en la aviación. Llegó a ser piloto mientras desarrollaba una técnica para visualizar mediante gráficos en 3D un vuelo simulado. Partiendo de esos estudios desarrolló el primer simulador de vuelo en 3D de la historia, *Flight Simulator* para Apple II, totalmente programado en código máquina. Llegó a fundar dos compañías de desarrollo de simuladores: SubLogic y BAO Ltd., que han desarrollado toda la serie *Flight Simulator* para todos los sistemas. Actualmente, BAO pertenece a Microsoft y SubLOGIC a Sierra, pero ArtWick sigue escribiendo libros y creando los más complejos simuladores para PC.



Bruce ArtWick

SIMULADORES DE VUELO DE COMBATE

Otra alternativa de simulación propuesta y que atiende más a la jugabilidad es la unión entre los aspectos aeronáuticos y la acción del combate. Casi toda la producción de simuladores de vuelos atiende a esta premisa, buscando no solo una simulación aérea, sino simular misiones de combates históricas o ficticias. Prácticamente, a lo largo de la historia, desde los primeros títulos de 8 bits como *Kennedy Approach* o *MiG Alley Ace* (primer simulador de combate multijugador en pantalla dividida), las diferencias entre los títulos publicados se basaban en la calidad de la representación gráfica de los combates. De estos primeros títulos, precursores de los simuladores de hoy, destacamos la serie *F-15 Strike Eagle*, *Spitfire Ace* o *Gunship* y *F-19 Stealth Fighter* de Microprose. Los simuladores actuales son exquisitos gráficamente y cada vez se desarrollan más títulos de calidad. Microsoft aparece en este tipo de simulación con *Combat Flight Simulator* 1 y 2. Pero, en virtud a sus cualidades, matizamos la fantástica producción *European Air War* de Microprose con un realismo aplastante con hasta 256 aviones simultáneos en combate o el original manejo de bombarderos B-17 en *B-17 Flying Fortress*. Este género siempre ha estado acompañado por cierta dificultad en el manejo del juego y que no ayudaba mucho a capturar adeptos ajenos a la simulación aérea. Sin embargo, algunas producciones se decantaban por una simulación aceptable, orientada a la acción inmediata y sin muchos entresijos en el control de los aviones. Desarrolladoras como Jane´s Combat Simulations optaron por esta fórmula y produjeron títulos de gran calidad como la serie *USAF* con gráficos detallados al milímetro de aviones y escenarios. Pero es en producciones de nivel superior orientada a pilotos expertos, donde encontramos títulos realmente sorprendentes como *EF2000* y de nuevo las producciones de Jane´s Combat Simulations como la serie *SkunkWorks*, *F-15* o el fantástico simulador del caza *F/A-18 Hornet*, *F/A-18 Simulator*. No podemos olvidar la saga *Falcon 4.0* de Micorprose con su exquisita IA y sus campañas en tiempo real. Tampoco queremos terminar sin mencionar la famosa saga de simuladores para helicópteros *Comanche* de Novalogic con más de 3 millones de copias vendidas.

SIMULADORES ON-LINE

Desde que internet llega a los hogares de todo el mundo, los simuladores de vuelo han entrado a formar parte también del juego On-Line. Al igual que ocurre con géneros como el FPS táctico o el RPG, aparecen títulos en donde el jugador participa en batallas aéreas ficticias o históricas. El primer simulador On-Line fue *Air Warrior*. Le siguió *War Birds* y actualmente destaca *Aces High* en el que también es posible manejar otros vehículos militares.

SIMULADORES DE VUELO ESPACIALES

Paralelo al desarrollo de simuladores de vuelos reales, aparecieron otros basados en el vuelo y combate espacial. El liderazgo de este tipo de simulación desde el comienzo de la década de los 90 ha sido para la serie *Wing Commander* (Origin) y *X-Wing* (Totally Games) El primero destacó por la inclusión de escenas narrativas en 3D durante el juego para dar más énfasis al guión. Por otro lado, *X-Wing* se basaba en la laureada serie de películas *Star Wars,* lo que le proporcionaba un éxito asegurado. Sin embargo, otros títulos aparecieron para intentar hacer sombra a estas series a partir de 1998.



En el próximo número...

... hablaremos de los simuladores deportivos.

Cuestionario Videojuegos 16

Preguntas

1. **Escribe las sentencias necesarias para leer y mostrar los ficheros de un directorio.**

2. **Escribe el código para abrir un fichero y grabar en él 10 números enteros aleatorios.**

3. **¿Cómo una entidad puede detectar la proximidad de otra?**

4. **¿Por qué es necesario incluir todas las variables que intervienen en el comportamiento de una entidad (contadores, switchers,etc.) en su estructura de tipo?**

5. **¿Qué dos formas podemos usar para disponer de un terreno en nuestro juego?**

6. **Si usamos Paint Shop Pro para dibujar el mapa de alturas, ¿cómo podemos visualizar el terreno resultante?**

7. **¿Cómo podemos importar un fichero de audio en formato .WAV en Anvil Studio?**

8. **¿Cuáles son los procedimientos a seguir para añadir una muestra a una pista de ritmos?**

9. **¿Cómo podemos obtener capturas de nuestro juego?**

10. **Antes de empezar a editar con Ulead MediaStudio, ¿qué es lo primero que debemos hacer?**

Respuestas al cuestionario 15

▷ **1.** While Not KeyHit (16)
PROCESOS
Wend

▷ **2.** Mediante la instrucción MouseZ().
El valor devuelto por MouseZ() se incrementa cuando la rueda se desliza hacia delante y decrementa cuando lo hace hacia atrás.

▷ **3.** Asignando coordenadas X y Z aleatorias dentro de un rango determinado:
CoordX= Rand (RangoX1, RangoX2)
CoordZ= Rand (RangoZ1, RangoZ2)
PositionEntity Entidad, CoordX, Altura Terreno, CoordY

▷ **4.** Comprobando que todos los datos leídos tienen valor 0 o "". O bien utilizando la instrucción Eof:
While Not Eof (Fichero)

▷ **5.** Para crear un cubo en Bryce, solo tenemos que pulsar en el icono con el dibujo de un cubo situado en el menú *Create*. Para situarlo frente a la cámara podemos hacerlo con las herramientas de edición del menú *Edit* desplazando el cubo, o bien moviendo la cámara con los iconos de navegación.

▷ **6.** Para asignar y editar una textura a un modelo en Bryce debemos entrar en el laboratorio de materiales pulsando sobre la letra "M" situada en la parte inferior derecha del modelo.

▷ **7.** Para seleccionar un grupo de notas en la sección compose de Anvil Studio debemos hacerlo sobre la línea de tiempo.

▷ **8.** En primer lugar elegimos la calidad de la muestra y posteriormente pulsamos sobre el botón "VU". Para terminar hacemos *Play* en el CD y pulsamos el botón "REC".

▷ **9.** Grabando los datos en matrices o bien utilizando bancos de memoria.

▷ **10.** Utilizando bancos de memoria y usando las instrucciones:
PokeByte, PokeShort, PokeInt o PokeFloat para escribir en memoria y:
PeekByte, PeekShort, PeekInt o PeekFloat para leer.

Contenido CD-ROM 16

▶ AUDIO

■ Audiotest

Chequea rápidamente el audio de tu PC antes de empezar a editar y manipular ficheros de audio.

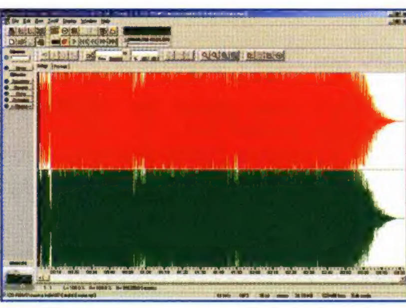
■ Mkw Audio Compression Tool

Ahorra espacio comprimiendo tus archivos de audio.

■ Total Recorder

Graba sonidos y música usando esta aplicación, que viene junto a los drivers necesarios.

■ WavePurity 4.10



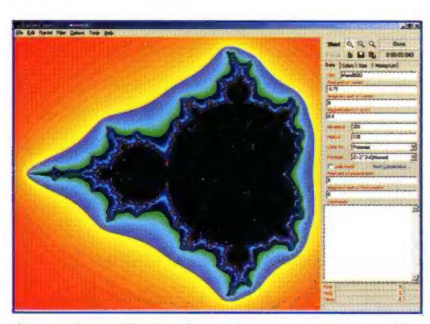
Editor muy completo de audio que te será de gran ayuda para tus ficheros de música.

▶ DISEÑO 2D

■ Antares 8

Avanzado navegador y visualizador de imágenes diseñado específicamente para manejarlas cómoda y fácilmente.

■ FractalForge



Aprende a dibujar fractales que podrás utilizar para tus fondos y paisajes psicodélicos.

■ ImageKeep Express 1.0.1

Crea bases de datos con tus imágenes y así tenlas siempre ordenadas y a mano.

■ Magic Converter 3.0

Con esta utilísima aplicación podrás cambiarle el formato a numerosos grupos de imágenes.

■ PFS Manager 2.05

Maneja tus álbumes fotográficos y ten ordenadas tus imágenes en sus categorías correspondientes.

▶ DISEÑO 3D

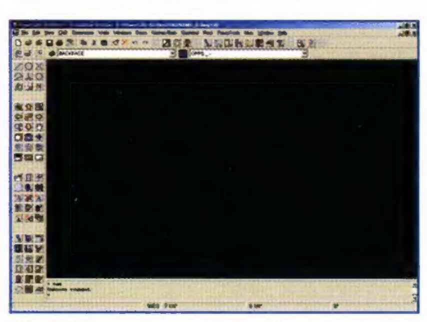
■ 3D Coder

Rápida aplicación para realizar pequeños ajustes en las imágenes en tres dimensiones.

■ 3D RealTime Visualizer 1.0

Crea objetos en 3D rápidamente y sin esfuerzo.

■ PowerCAD Architect



Magnífico programa de arquitectura para crear interiores en 3D, edificios y todo lo que se te ocurra.

■ Wings 3D

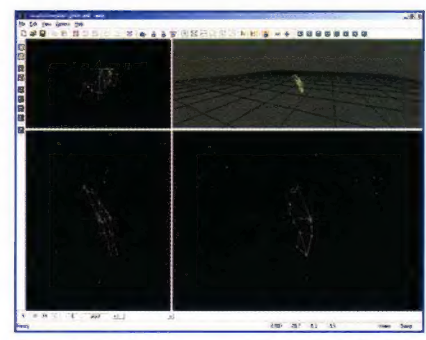
Sencillo editor de imágenes en tres dimensiones con una interfaz muy intuitiva.

▶ PROGRAMACIÓN

■ ActiveZipper Pro 1.0

Herramienta para Visual Basic para comprimir y descomprimir fácilmente.

■ 3D GameStudio A4



Completa suite para programar y renderizar juegos en 3D.

■ Pocket C

Editor de código "de bolsillo", para ver rápidamente cómo van tus líneas de código.

▶ JUEGOS

■ Falcon 4.0

Simulador de combates aéreos con un gran concepto de la inteligencia artificial y con campañas en tiempo real.

■ FreeSpace

Simulador de vuelos espaciales, surca los cielos gracias a este excelente juego.

■ X-Plane 6.4



Magnífico simulador de vuelo con el que te creerás pilotando de verdad.

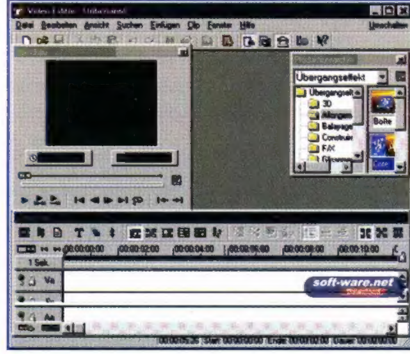
■ Zone of fighters

Como todas las semanas, nuestro juego tal y como va quedando.

▶ VÍDEO

■ Easy Video Joiner 4.01

Con esta útil aplicación podrás unir fácilmente varios clips de película en uno solo.



■ Ulead Media Studio Pro 6.5

Nueva oportunidad de conseguir este excelente software.

▶ EXTRAS

En este apartado encontrarás todos los ejemplos de los que hablamos en el coleccionable, para que no pierdas detalle.